

WEB

第 8 章 数据库编程

本章目录

- 8.1 JDBC基础
- 8.2 MySQL的使用
- 8.3 JDBC编程示例
- 8.4 数据库连接池技术
- 8.5 Web数据库编程
- 8.6 本章小结

前言

在当今互联网上各种丰富多彩的 Web 应用中，几乎无一例外地都要与数据库打交道，与数据库的交互是 Web 应用中必不可少的环节。

前言

以某论坛的应用为例：



主题：					
成功建立 jsp开发环境 [1 2 3 4 5]	yankov	95	2781	07-12-23 10:44:36	horizonNow
jsp开发问题	晨辉	2	258	07-12-8 19:46:05	weizhongqin
[原创]我自己安装的一个软件出错了，谁能告诉这是什么意思吗	cdplzg8044	0	105	07-12-4 13:47:28	
解析神秘的百度蜘蛛! [原创] [1 2]	g14d	27	1252	07-11-30 15:22:58	shaoml
使用request.getProtocol()要导入哪些包呀？	风行柳飘	1	269	07-11-16 15:36:29	风行柳飘
如何释放JSP中的变量呀	风行柳飘	5	271	07-11-15 16:38:42	风行柳飘
[原创]北京诚聘jsp程序员	omomx	0	226	07-11-14 9:51:46	
IIS6与Tomcat6 整合笔记	Aimeko	4	598	07-11-9 9:15:47	Aimeko
关于JDBC5.1.5的用法	风行柳飘	3	275	07-10-29 19:33:50	redmoon
关于CWBS2.3RC1 的一点疑惑[求助]	风行柳飘	1	239	07-10-29 14:31:38	redmoon
[公告]承接网站建设,web应用开发:www.131hr.com	springfall	0	259	07-10-28 21:23:29	
[求助]各位大侠，帮我推荐几本JSP教材吧	风行柳飘	0	242	07-10-28 18:04:18	
[原创]富文本怎么搞	xsn	2	376	07-10-9 13:26:23	milan
勿讲	tdz	0	364	07-9-24 18:09:19	

浏览者要在论坛中发帖子，必须先注册成为该论坛的用户

前言



The screenshot shows a web browser window with the address bar displaying `http://www.cloudwebsoft.com/regist.jsp`. The page title is "注册 - 云网论坛 - Powered by CWBBS JSP论坛". The main content area is titled "CWBBS" and contains a registration form. The form includes fields for "用户名" (Username), "密码" (Password), "性别" (Gender), "Email", and "问题" (Question). There are also checkboxes for "注意：此栏必须填写" (Attention: This field must be filled) and "将以下带*号信息保密" (Keep the following information with * confidential). The form is submitted by clicking the "确定" (Confirm) button. The footer of the page contains the text "Powered by CWBBS 2.3RC1 © 2005-2007 Cloud Web Soft Gzip enabled wap2.0" and contact information: "站务联系：QQ: 51066962 Email: welcome@cloudwebsoft.com 苏ICP备05064076号".

注册 - 云网论坛 - Powered by CWBBS JSP论坛|博客|CMS|文章管理|办公软件|OA|威客|换客|掘金|商...

http://www.cloudwebsoft.com/regist.jsp

Google

注册 - 云网论坛 - Powered by CWBBS JSP论坛...

CWBBS

首页 论坛 登录 用户中心 论坛状态 会员 社区服务 风格 分栏模式 博客 朋友圈

会员代号及密码 注意：此栏必须填写 请等待管理员审核通过您的帐户

用户名 * 检查用户名

密码 * 确认密码 *

性别 ☒ 男 ☐ 女

Email *

问题 答案

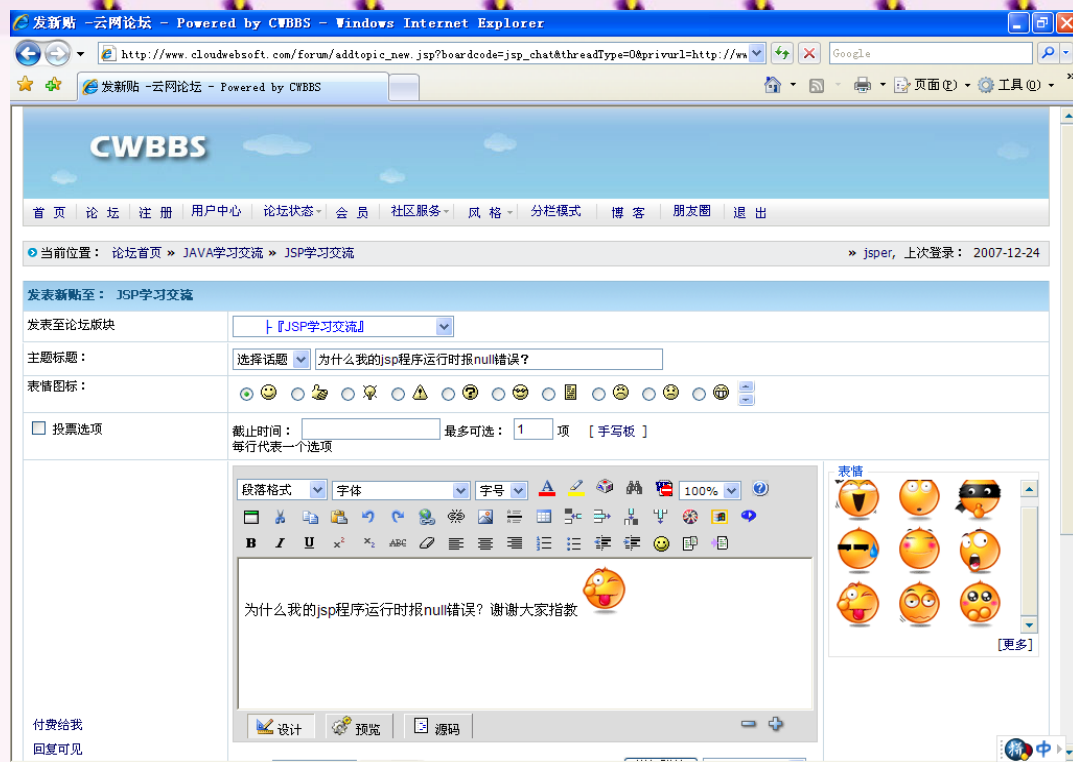
+ 点击此处 填写详细资料 ☐ 将以下带*号信息保密

确定 重置

Powered by CWBBS 2.3RC1 © 2005-2007 Cloud Web Soft Gzip enabled wap2.0
站务联系：QQ: 51066962 Email: welcome@cloudwebsoft.com
苏ICP备05064076号

用户填写注册表单后单击【确定】按钮把个人注册信息发送给服务器，由服务器端的程序将这些数据保存到数据库里。

前言



用户发表一篇帖子，其实是将这篇帖子的标题、正文等内容存到了数据库里；而其他用户回复某一篇帖子，其实也是对数据库执行了一次新纪录插入的过程。

前言

数据库编程技术是我们在 Web 项目中的不可或缺的开发技术。本章内容主要是以 JDBC 编程技术为基础，介绍一些常见的数据库编程接口以及数据库连接池技术，并结合主流的网络数据库管理系统 MySQL 进行实例开发。

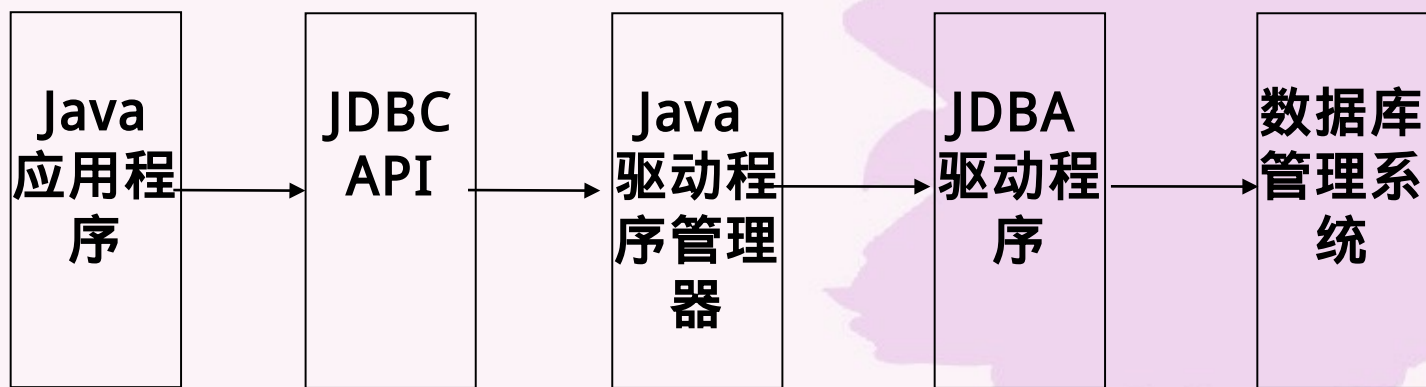
8.1 JDBC 基础

JDBC (Java Database Connectivity) 是 Sun 公司提供的一套数据库编程接口 API (Application Programming Interface) 函数，它由一组用 Java 编程语言编写的类和接口组成。

Java 基础包中的 java.sql 下面的 DriverManager、Connection、Statement、ResultSet 等类和接口

8.1JDBC 基础

JDBC 工作流程图



8.1.1 JDBC 驱动程序

数据库厂商必须提供相应的驱动程序并实现 JDBC API 所要求的基本接口，即每个数据库系统厂商必须提供对 Connection、Statement、ResultSet 等接口具体实现，从而最终保证 Java 程序员通过 JDBC 实现对不同的数据库操作。

8.1.1 JDBC 驱动程序

目前所使用的 JDBC 驱动程序可分为以下 4 类

- (1) JDBC-ODBC 桥加 ODBC 驱动程序：利用微软的 ODBC 驱动程序提供 JDBC 访问。
- (2) 本地 API – 部分用 Java 来编写的驱动程序
- (3) JDBC 网络纯 Java 驱动程序：这种驱动程序将 JDBC 转换为与 DBMS 无关的网络协议，之后这种协议又被某个服务器转换为一种 DBMS 协议。
- 4) 本地协议纯 Java 驱动程序：这种类型的驱动程序将 JDBC 调用直接转换为 DBMS 所使用的网络协议。

8.1.1 JDBC API

Java 程序员通过调用 JDBC API 实现连接数据库、执行 SQL 语句并返回结果集等编程数据库的能力，它主要是由一系列的接口和类定义所构成。如：DriverManager、Connection、Statement、PreparedStatement、ResultSet。

DriverManager

DriverManager 类是 JDBC 的管理层，作用于用户和驱动程序之间。它跟踪可用的驱动程序，并在数据库和相应驱动程序之间建立连接。

该类中最重要的是 `getConnection`，该方法将建立与数据库的连接。

当调用方法 `getConnection` 时，它将检查清单中的每个驱动程序，直到找到可以与指定数据库进行连接的驱动程序为止。

DriveManager

具体步骤：

调用方法 `Class.forName (String className)` 显式地加载驱动程序类

```
Class.forName("net.sourceforge.jtds.jdbc.Driver");
```

调用 `DriverManager.getConnection`，`DriverManager` 将检查每个驱动程序，查看它是否可以建立连接

```
Connection conn = DriverManager.getConnection("jdbc:jtds:sqlserver://localhost:1433;DatabaseName=shop","sa", "123");
```

Connection

Connection 接口是代表与数据库之间的连接，连接成功后可以对后台数据库执行 SQL 语句、返回处理结果。

Statement

Statement 对象用于将 SQL 语句发送到数据库管理系统中，作为在给定连接上执行 SQL 语句的包容器。

Statement

在建立了数据库连接 Connection 之后，就可用 Connection 的方法 createStatement 创建 Statement 对象

```
Statement stmt = con.createStatement();
```

Statement

Statement 接口提供了执行 SQL 语句的方法：`executeQuery` 和 `executeUpdate`。

Statement

`executeQuery` 方法用于查询结果集的语句，即 `SELECT` 语句，得到的结果是实现 `ResultSet` 接口的对象。

```
ResultSet rs = stmt.executeQuery("SELECT * FROM person");
```

`executeUpdate` 方法用于执行 `INSERT`、`UPDATE` 或 `DELETE` 等语句以及 `SQL DDL`（数据定义语言）语句

PreparedStatement

PreparedStatement 接口继承 Statement，但是，两者最大的区别在于 Statement 对象本身不包含 SQL 语句，而 PreparedStatement 包含已编译的 SQL 语句，所以其执行速度要快于 Statement 对象。

PreparedStatement

使用步骤：

1) 创建 PreparedStatement 对象

```
PreparedStatement pstmt = con.prepareStatement("UPDATE product SET p_name = ? , p_count = ? WHERE p_id = ? ");
```

PreparedStatement

2) 为 PreparedStatement 对象提供参数

```
pstmt.setString(1, " 手机 ");    // 商品名称
```

```
pstmt.setInt(2, 200);           // 商品数量
```

```
pstmt.setString (3, "20050102026"); // 商品编
```

码) 调用 PreparedStatement 对象的执行方法

```
pstmt.executeUpdate();
```

ResultSet

一个 ResultSet 对象对应着一个查询结果集，我们可以将这个结果集看成一个表。对 ResultSet 结果集中记录的访问必须逐行进行，常用 While 循环来处理。ResultSet 对象有一个指向当前行的指针，这个指针默认的初始位置指向第一行之前，ResultSet 的 next 方法使这个指针向下移动一行。因此，第一次使用 next 方法将指针指向结果集的第一行，这时可以对第一行的数据进行处理。

ResultSet

处理完毕后，继续使用 next 方法，指针移向下一行，继续处理第二行数据。next 方法的返回值是一个 boolean 型的值，该值若为 true，说明结果集中还存在下一条记录，并且指针已经成功指向该记录，可以对其进行处理；若返回值是 false，则说明没有下一行记录，结果集已经处理完毕。

° ResultSet 类的 getXXX() 方法可以从某一行中获得检索结果。其中 XXX 是 JDBC 中的 Java 数据类型，如 int、String、Date 等。

ResultSet

以下代码是查询商品表中所有商品的商品名称和商品数量记录

```
PreparedStatement pstmt = conn.prepareStatement("SELECT  
p_name, p_count FROM product");  
ResultSet rs = pstmt.executeQuery();  
while (rs.next()) {  
    String name = rs.getString("p_name");  
    int count = rs.getInt("p_count");  
    System.out.println(" 商品名称 : "+ name +" 数量 : "+ c  
ount);  
}
```

ResultSet

Sun 公司提供的 `getXXX` 提供两种方法对指定列进行检索：一种是以列名（`String` 对象），另一种是以列的索引（`int` 值）来检索。注意列的索引是从左至右编号，并且从列 1 开始。

```
String name = rs.getString(1);  
int count = rs.getInt(2);
```

ResultSet



小提示：
Statement 或 PreparedStatement
每次执行 SQL 语句都会关闭上一次
得到的 ResultSet 结果集。因此，
在重新执行 SQL 语句之前，请先完
成对上次得到的 ResultSet 对象的
处理。

8.2 MySQL 的使用

MySQL 是一个小型关系型数据库管理系统，
开发者为瑞典 MySQL AB 公司。

MySQL 同时也是一个典型的客户机 / 服务器
架构的网络数据库管理系统。

8.2.1 MySQL 服务器的安装配置

打开 MySQL 安装文件 MySQL-5.0.18-win32.zip，双击解压缩，运行“setup.exe”

8.2.1 MySQL 服务器的安装配置



小提示：

有些情况下会出现“Start service”出错，这可能是由于本机以前曾经安装过 MySQL，请先确认以前安装的 MySQL 是否彻底卸载掉了（包括原先 MySQL 文件夹下的一些配置文件，如 data 目录下的 ibdata、ib_logfile0、ib_logfile1 等），然后再重新安装。

8.2.2 MySQL 数据库的创建

MySQL 官方提供了两个客户端工具— MySQL Administrator 和 MySQL Query Browser，它们均提供了图形界面帮助我们对 MySQL 进行服务器管理或数据操作。

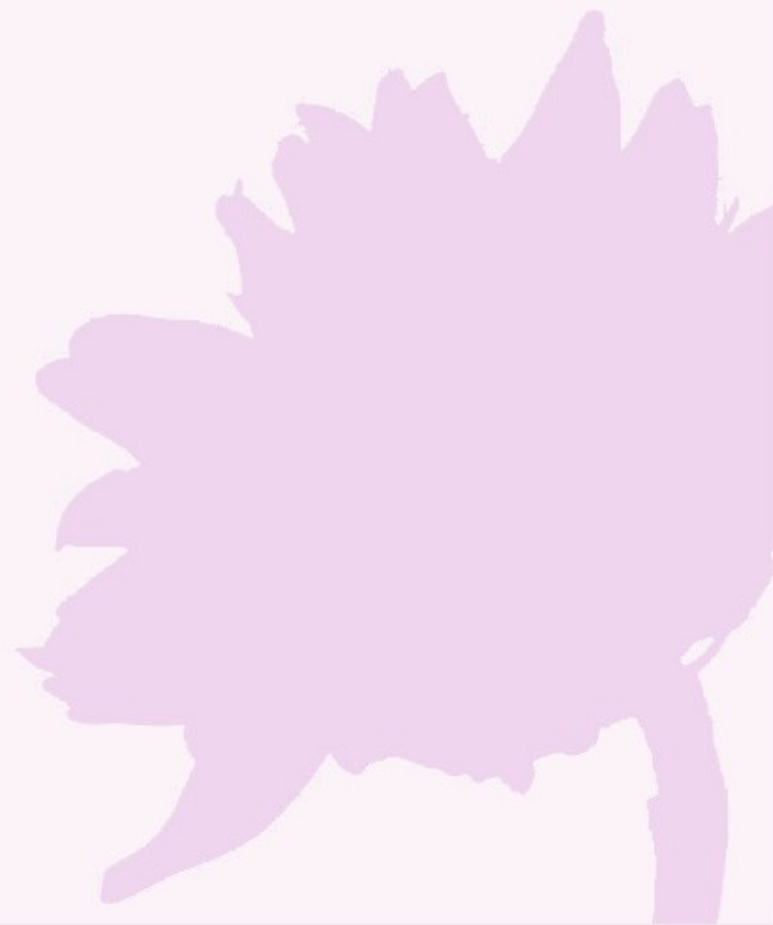
8.2.3 MySQL 数据库关系表的创建

MySQL Query Browser 也是 MySQL 官方的客户端工具之一，我们可以用 MySQL Query Browser 连接 MySQL 数据库，执行各种 SQL 语句。

8.2.4 MySQL 数据库的备份与还原

在 MySQL 中，数据库的备份与还原都是通过 MySQL Administrator 来完成的，数据备份的文件形式是 *.sql 文件。

8.3 JDBC 编程示例



8.3.1 运行环境配置

1. 新建 Java Project

首先，打开 MyEclipse ，新建一个 Project ，选择的项目类型为“Java Project”，单击【Next】按钮，弹出在弹出如图 8-40 所示窗体，在窗体中输入项目名称，单击【Finsh】按钮生成项目。

8.3.1 运行环境配置

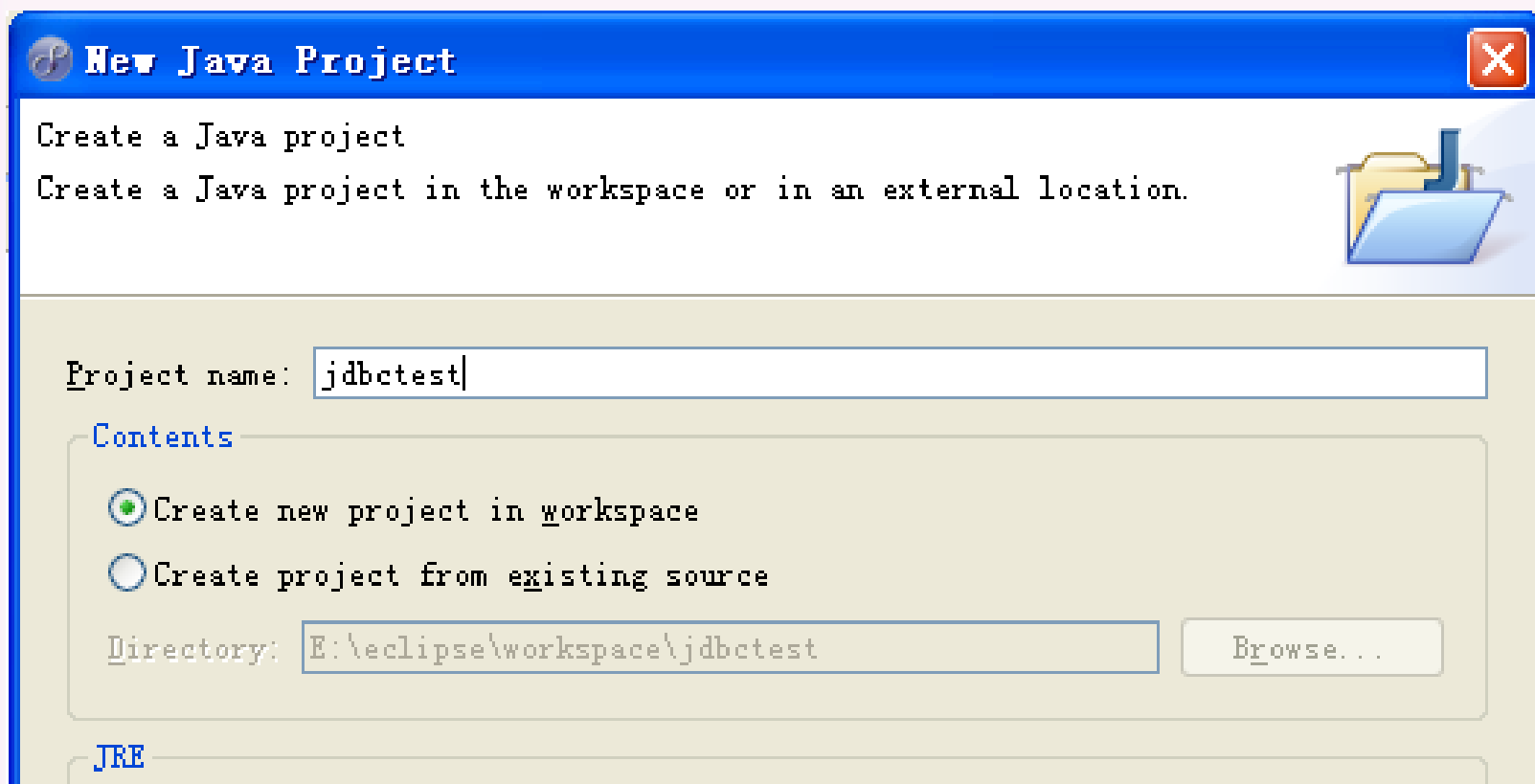


图 8-40 输入项目名称

8.3.1 运行环境配置

2. 加载 JDBC 驱动

在 8.1.1 中，我们已经介绍了 4 类 JDBC 驱动程序，这里我们采用第 4 类驱动，即 MySQL 官方发布的专用的驱动文件 `mysql-connector-java-5.1.0-bin.jar`（本书配套光盘 DB 目录下提供），将其拷贝到当前的 Project 中，如图 8-41 所示。

8.3.1 运行环境配置

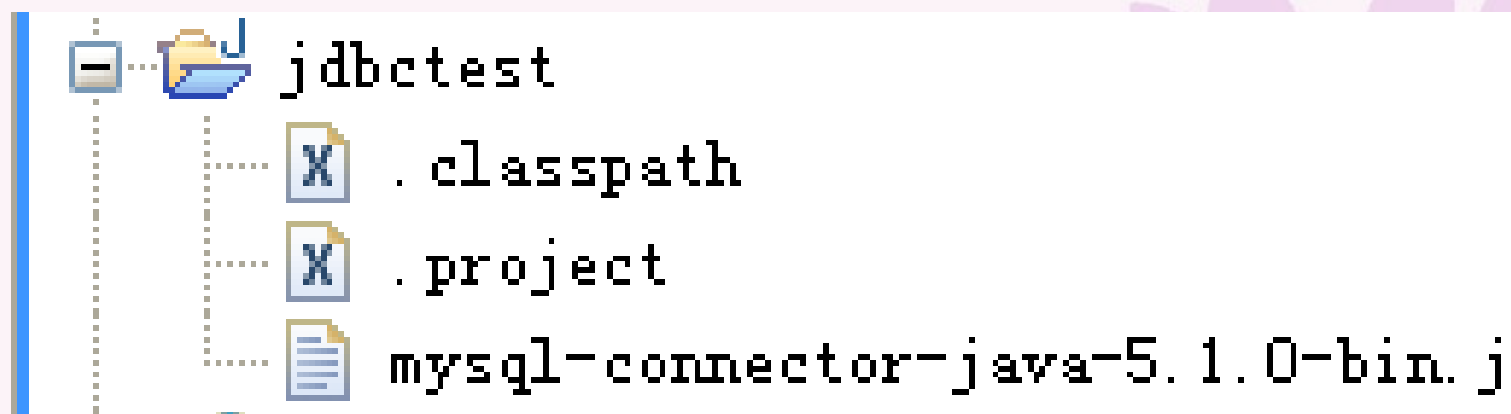


图 8-41 驱动文件

8.3.1 运行环境配置

2. 设置项目属性

右键选中该项目，在弹出菜单中选择【 Properties 》，对该项目的属性进行设置，如图 8-42 所示。

8.3.1 运行环境配置



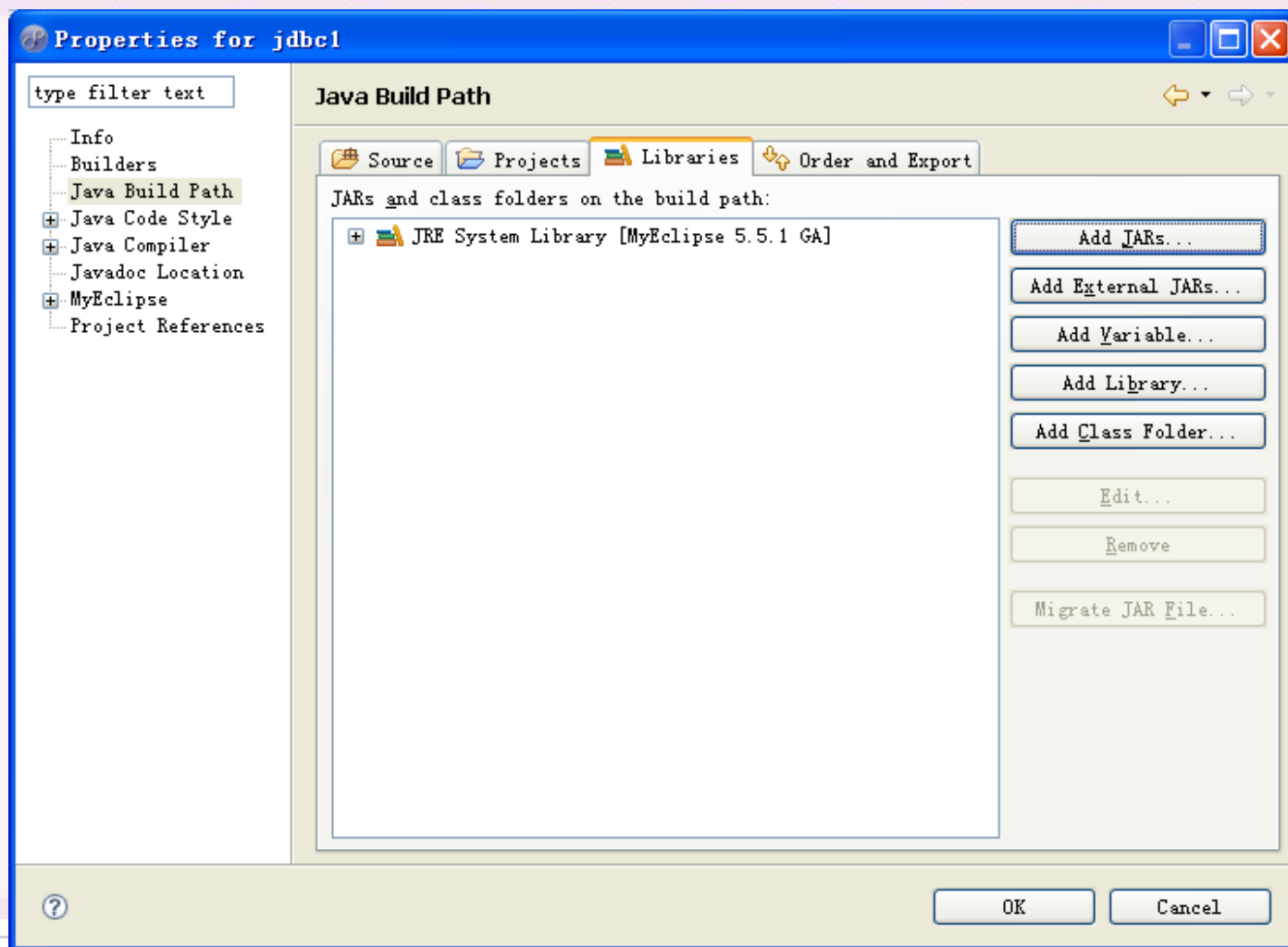
Run As ▶
Debug As ▶
Profile As ▶
Team ▶
Compare With ▶
Restore from Local History...
Synchronize ▶
MyEclipse ▶
PDE Tools ▶
Properties

图 8-42 设置项目属性

8.3.1 运行环境配置

如图 8-43 所示，在项目的属性窗体中，点击左侧菜单中的【Java Build Path】进行编辑。在右侧的标签卡中，我们选择【Libraries】，对当前项目所用到的 Java 类库进行设置，单击【Add JARs】按钮添加所需要的 Jar 包文件。

8.3.1 运行环境配置



8.3.1 运行环境配置

在弹出窗体中，选择当前项目中的驱动程序 `mysql-connector-java-5.1.0-bin.jar`，如图 8-44 所示。

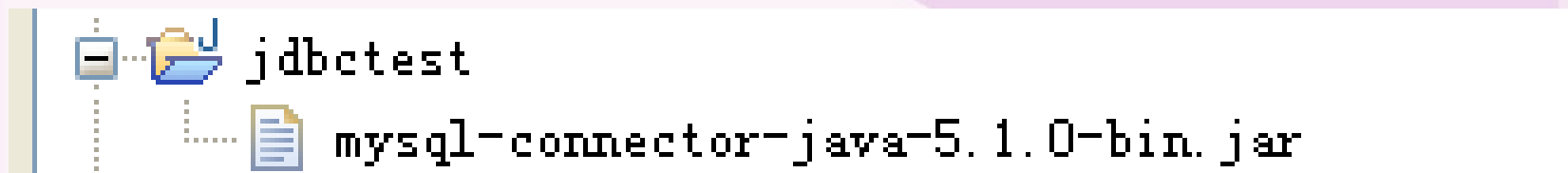
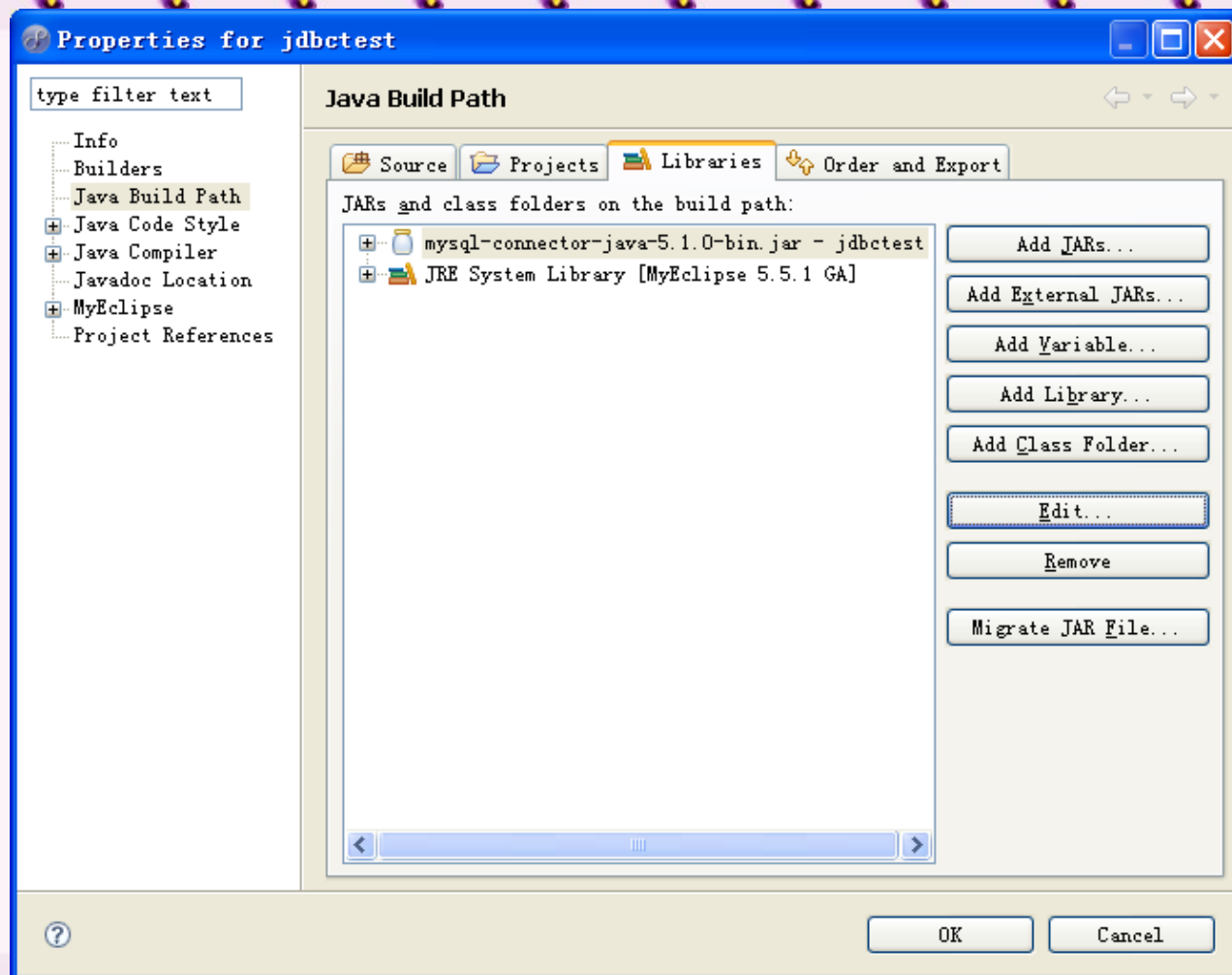


图 8-44 选择驱动程序

8.3.1 运行环境配置

添加成功后，可以看到 Libraries 中出现了该 JAR 包，如图 8-45 所示。

8.3.1 运行环境配置



8.3.2 查询操作

在当前的项目中创建一个新的 Class 类文件，该类名为 dbquery，功能为查询并打印 dbtest 数据库的 person 表中 id 值为 100 的人员信息。实现代码如下：

8.3.2 查询操作

```
import java.sql.*;
public class dbquery {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306
/dbtest?
useUnicode=true&characterEncoding=gbk","root",
"test");
            String strSql="select * from person where id=? ";
            PreparedStatement pstmt =
conn.prepareStatement(strsql,
ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATAB
LE);
            pstmt.setInt(1, 100);
            ResultSet rs = pstmt.executeQuery();
```

8.3.2 查询操作

```
while (rs.next()) {  
    int id=rs.getInt("id");  
    String name = rs.getString("name");  
    int age=rs.getInt("age");  
    System.out.println(" 编号: "+id+" 姓名: " +name+" 年  
龄: "+age);  
}  
rs.close();  
pstmt.close();  
conn.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}  
}
```


8.3.2 查询操作

在以上代码中，创建 PreparedStatement 对象时使用了带 3 个参数的方法，采用这种方式创建的 PreparedStatement 执行查询 SQL 语句后，得到的结果集中的游标可以上下自由滚动，这将是下一章中对结果集进行分页显示处理的前提。

8.3.2 查询操作

此外，作为一种良好的编程风格，数据操作完毕后，应依次关闭 ResultSet、PreparedStatement 和 Connection 对象，这将立即释放本次数据库访问所占用的资源，有助于避免潜在的内存问题。

8.3.3 插入操作

在当前的项目中创建一个新的 Class 类文件，该类名为 dbinsert，功能为插入一条新纪录到 dbtest 数据库的 person 表中，查询并打印该表所有的记录。实现代码如下：

8.3.3 插入操作

```
import java.sql.*;
public class dbinsert {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306
/dbtest?
useUnicode=true&characterEncoding=gbk","root",
"test");
            String strsql = "insert into person(name,age)
values(?,?)";
            PreparedStatement pstmt =
conn.prepareStatement(strsql);
            pstmt.setString(1, " 李晓华 ");
            pstmt.setInt(2, 22);
            pstmt.executeUpdate();
        }
    }
}
```

8.3.3 插入操作

```
    strsql = "select * from person";
    PreparedStatement qpstmt = conn.prepareStatement(strsql,
ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
        ResultSet rs = qpstmt.executeQuery();
    while (rs.next()) {
        int id=rs.getInt("id");
        String name = rs.getString("name");
        int age=rs.getInt("age");
        System.out.println(" 编号: "+id+"  姓名: " +name+"  年
龄: "+age);
    }
    rs.close();
    pstmt.close();
    conn.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

8.3.3 插入操作

运行程序，从控制台输出的结果我们可以看到多出一条记录：编号 101、姓名“李晓华”，年龄 22 岁，表明插入成功。以上代码中，实现了两个功能，第一部分执行插入，第二部分进行查询和打印。

8.3.4 更新操作

更新操作的开发与插入相类似，只需修改 sql 语句和相应的参数。假设我们要将编号为 100 的人员的年龄更新为 24 岁，核心的代码如下：

8.3.4 更新操作

```
String strSql = "update person set age=?  
where id=? ";  
PreparedStatement pstmt =  
conn.prepareStatement(strsql);  
pstmt.setInt(1, 24);  
pstmt.setInt(2, 100);
```

运行程序，从控制台输出的结果我们可以看到编号为 100 的“陈刚”年龄为 24 岁，表明更新成功。

8.3.5 删除操作

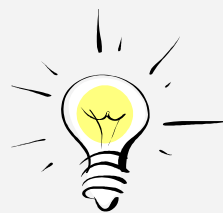
删除操作的开发也与前面的插入和更新相类似，只需修改 sql 语句和相应的参数。假设我们要将编号为 100 的人员记录从 person 表里删除，核心的代码如下（也可直接替换插入代码中的粗体字部分）：

8.3.5 删除操作

```
String strSql = "delete from person where  
id=? ";  
PreparedStatement pstmt =  
conn.prepareStatement(strsql);  
pstmt.setInt(1,100);
```

运行程序，从控制台输出的结果我们可以看到已经没有了 id 值为 100 的记录，表明删除成功。

小提示



小提示：

在进行插入、更新和删除操作的时候，我们采用的是 PreparedStatement 的 executeUpdate() 方法，而进行查询的时候则是用 executeQuery() 方法，后者返回结果集（ ResultSet ），注意区别。

8.4 数据库连接池技术

在使用 JDBC 进行与数据库有关的应用开发中，过多的数据库连接或数据库连接管理混乱所造成的服务器资源开销过大成为制约大型企业级应用的瓶颈。对数据库连接的管理能显著影响到整个应用程序的伸缩性和健壮性，影响到程序的性能指标。

8.4 数据库连接池技术

数据库连接池正是针对这个问题提出来的。对于众多用户访问的 Web 应用，采用数据库连接池技术在效率和稳定性上比采用传统的连接模式要优化很多。

8.4 数据库连接池技术

如图 8-46 所示，展示了采用了连接池技术对数据库进行管理和控制的流程图。

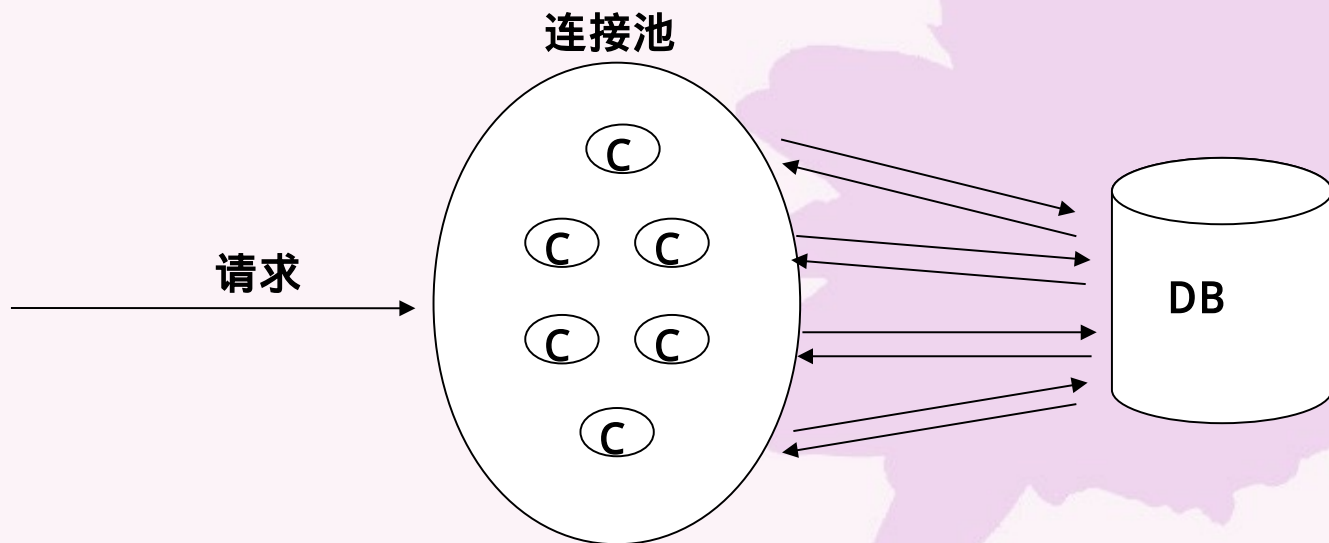


图 8-46 采用连接池技术对数据库进行管理和控制的流程图

8.4.1 连接池的工作原理

数据库连接池负责分配、管理和释放数据库连接（ Connection ），它最大的优点是允许应用程序重复使用现有的数据库连接，而不需要总是创建新的连接。连接池的核心工作原理如下：

8.4.1 连接池的工作原理

1. 连接池在初始化时创建一定数量的数据库连接放到连接池中，这里的“一定数量”是指连接池的最小连接数。无论这些数据库连接是否被应用程序所使用，连接池都将一直保证至少有这么多的连接处于连接池中。

8.4.1 连接池的工作原理

2. 当应用程序向数据库连接池发送连接请求时，只要连接池中有处于空闲状态的连接，就可以直接取出来使用。如果超过最小连接数了，连接池会创建新的连接。这些大于最小连接数的数据库连接在使用完不会马上被释放，它将被放到连接池中等待重复使用或是空闲超时后被释放。

8.4.1 连接池的工作原理

3. 当应用程序向连接池请求的连接已经超过了最大连接数时，这些请求将被加入到等待队列中，直到前面有使用完的连接被放回到连接池里，处于空闲状态，才可以继续使用这些连接。

8.4.1 连接池的工作原理

4. 当数据库连接的空闲时间超过最大空闲时间，则由连接池来彻底释放这些连接，以此来避免因为没有释放而引起的数据库连接遗漏。这项技术能明显提高对数据库操作的性能。

8.4.1 连接池的工作原理

在设置连接池的最小连接数和最大连接数的设置要考虑到下列因素：

8.4.1 连接池的工作原理

1. 最小连接数是连接池一直保持的数据库连接数，如果设置过大，而应用程序对数据库连接的使用量很小，那么将会有较多的数据库连接资源被浪费；如果设置过小，远小于日常的访问量，那么连接池将要频繁地创建新连接，浪费服务器资源。

8.4.1 连接池的工作原理

2. 最大连接数是连接池能容纳连接的最大数目，如果设置过小，而数据库连接请求数经常超过这个数目，那么将会经常有请求被加入到等待队列中，这会影响整个应用系统的访问效率；如果设置过大，超过服务器内存的负荷，反而可能导致服务器宕机。

8.4.2 连接池的配置与应用

Proxool 是一种业界公认的性能较好的连接池实现技术，它提供了对常见的数据库驱动程序的连接池封装，只需要进行简易的配置就可以移植到现有的代码中，为现有的 JDBC 驱动程序增加连接池功能。

8.4.2 连接池的配置与应用

1. 下载 proxool 的 zip 压缩包
2. 解压缩 proxool , 复制其中 lib 下面的 proxool-0.9.jar 到 Web 项目中的 web-info/lib 下。
3. 根据当前使用的数据库类型将其 jdbc 驱动的 jar 包复制到 web-info/lib 下。
4. 在 web-info 下建立文件 : proxool.xml , 文件内容如下 :

8.4.2 连接池的配置与应用

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<something-else-entirely>
  <proxool>
    <alias>MySQL</alias>
    <driver-url>
      jdbc:MySQL://localhost:3306/dbtest?
      useUnicode=true&characterEncoding=gb2312</driver-url>
    <driver-class>com.MySQL.jdbc.Driver</driver-class>
    <driver-properties>
      <property name="user" value="root" />
      <property name="password" value="mis" />
    </driver-properties>
    <maximum-connection-count>5000</maximum-connection-count>
    <minimum-connection-count>5</minimum-connection-count>
    <maximum-active-time>60000</maximum-active-time>
  </proxool>
</something-else-entirely>
```

8.4.2 连接池的配置与应用

5. 在 web.xml 文件内进行如下配置，使当前的 Web 项目启动时就读取 proxool.xml 中的各项配置信息从而进行连接池的创建。如下图：

8.4.2 连接池的配置与应用

```
<servlet>
  <servlet-name>ServletConfigurator</servlet-name>
  <servlet-class>
org.logicalcobwebs.proxool.configuration.ServletConfig
urator
</servlet-class>
  <init-param>
    <param-name>xmlFile</param-name>
    <param-value>WEB-INF/proxool.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

8.4.2 连接池的配置与应用

6. 如果需要查看连接池实时的运行情况，可以继续往 `web.xml` 中添加配置代码，使用 `proxool` 提供的管理监控工具类 `AdminServlet`，访问路径为 `/admin`。

8.4.2 连接池的配置与应用

```
<servlet>
  <servlet-name>Admin</servlet-name>
  <servlet-class>
org.logicalcobwebs.proxool.admin.servlet.AdminServ
let</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Admin</servlet-name>
  <url-pattern>/admin</url-pattern>
</servlet-mapping>
```

8.4.2 连接池的配置与应用

7. 如果想测试连接池是否运行成功，可以编写一个JSP文件，在页面中嵌入调用数据库连接的代码片段：

8.4.2 连接池的配置与应用

```
<%!int i=0;%>
<%
    try {

Class.forName("org.logicalcobwebs.proxool.ProxoolDriver");
        Connection conn =
DriverManager.getConnection("proxool.MySQL");
        i++;
        out.print(" 您已获取 "+i+" 个数据库连接 ");
    } catch (Exception e) {
        e.printStackTrace();
    }
%>
```

以上代码中的“proxool.MySQL”中的“MySQL”必须与proxool.xml中的alias（连接池别名）值对应。

8.4.2 连接池的配置与应用

8. 在地址栏里访问以上 JSP，每次访问即获取连接池中的一个连接，然后访问 <http://localhost:8080/dbpooltest/admin> 查看连接池的运行状况，如图所示。

8.4.2 连接池的配置与应用

Proxyool Admin - Windows Internet Explorer

http://localhost:8080/dbpooltest/admin

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

Proxyool Admin

Proxyool 0.9.0RC3 (10-Jan-2007 01:38)

Definition Snapshot

alias:	mysql
driver-url:	jdbc:mysql://localhost:3306/dbtest?useUnicode=true&characterEncoding=gb2312
driver-class:	com.mysql.jdbc.Driver
minimum-connection-count:	5
maximum-connection-count:	5000
prototype-count:	-
simultaneous-build-throttle:	10
maximum-connection-lifetime:	04:00:00
maximum-active-time:	00:01:00
house-keeping-sleep-time:	30s
house-keeping-test-sql:	-
test-before-use:	false
test-after-use:	false
recently-started-threshold:	00:01:00
overload-without-refusal-lifetime:	00:01:00
injectable-connection-interface:	-

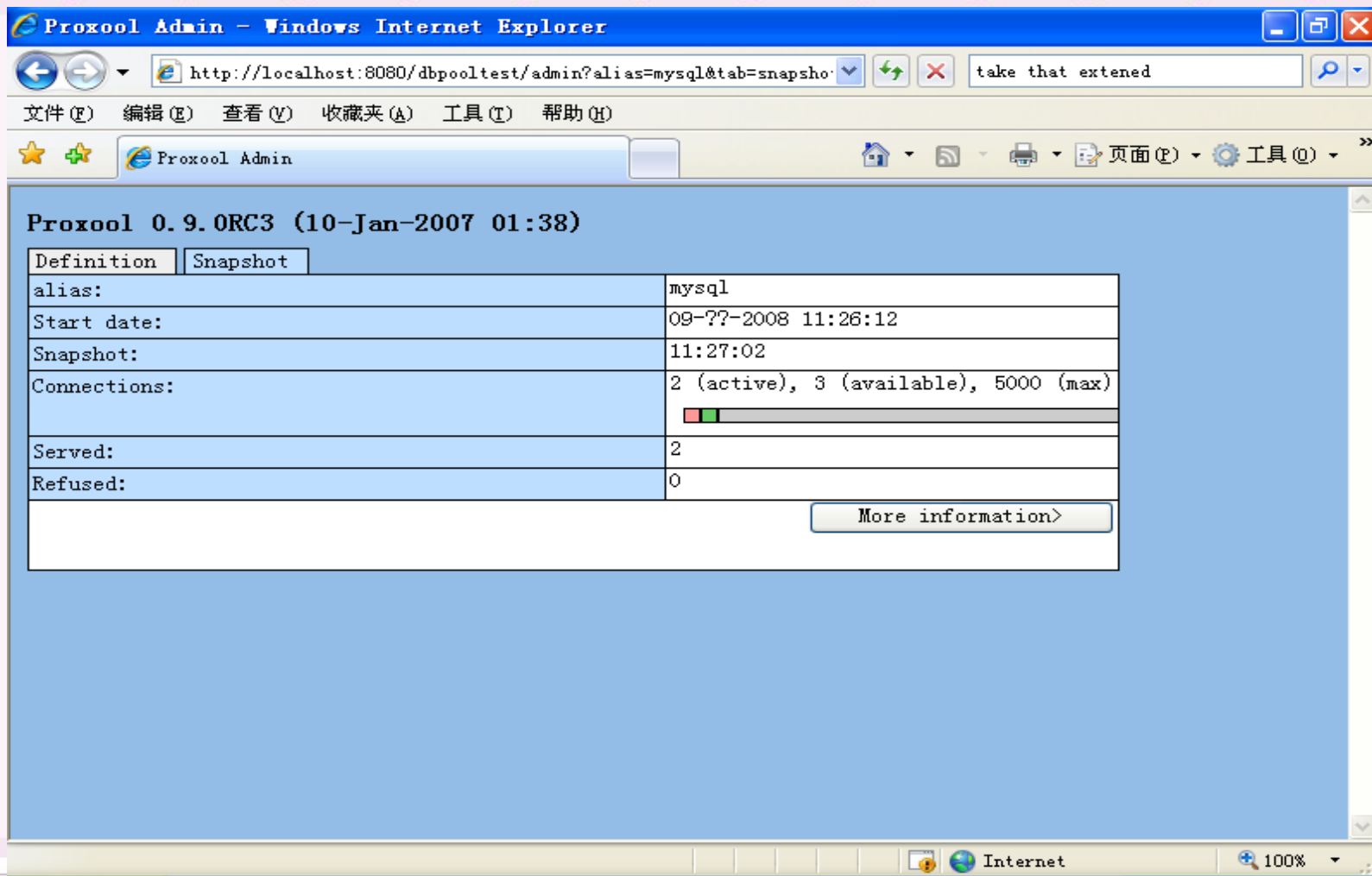
http://localhost:8080/dbpooltest/admin?alias=mysql&tab=snapshot

Internet 100%

8.4.2 连接池的配置与应用

单击【 Snapshot 】按钮就可以看到目前连接池中的正被占用的活动连接数（ Active ）、处于空闲状态的可用连接数（ available ）以及所容纳的最大连接数（ max ），如图 7-43 所示。

8.4.2 连接池的配置与应用




Proxool Admin - Windows Internet Explorer

http://localhost:8080/dbpooltest/admin?alias=mysql&tab=snapsho

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

Proxool Admin

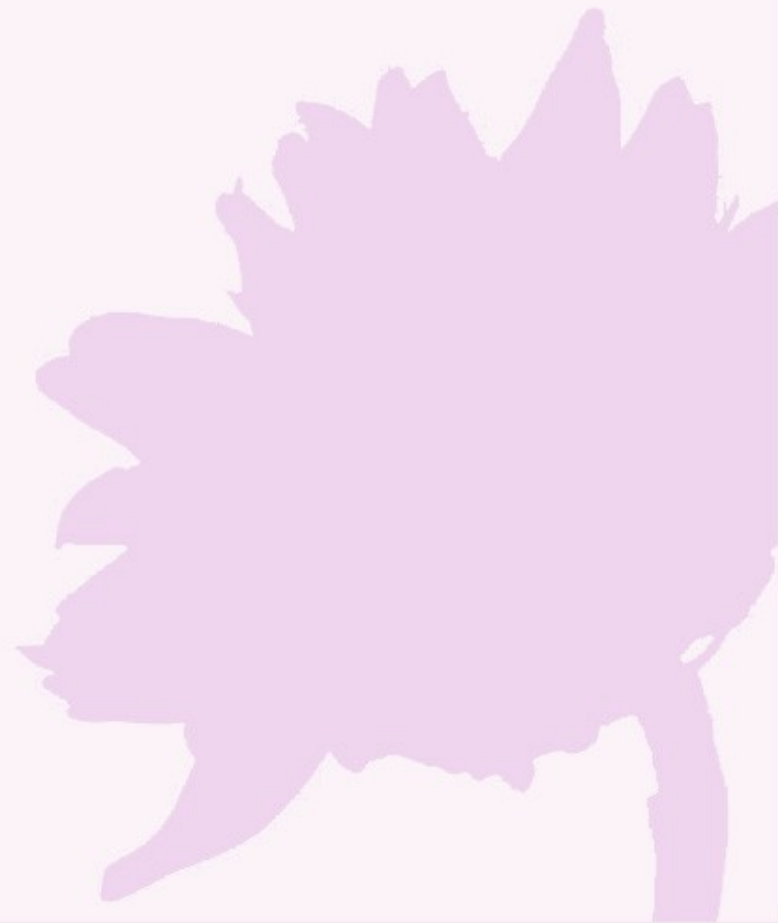
Proxool 0.9.0RC3 (10-Jan-2007 01:38)

Definition	Snapshot
alias:	mysql
Start date:	09-??-2008 11:26:12
Snapshot:	11:27:02
Connections:	2 (active), 3 (available), 5000 (max) 
Served:	2
Refused:	0

[More information>](#)

Internet 100%

8.5 Web 数据库编程



8.5.1 数据库连接对象设计

在 Web 项目中，数据库编程的代码属于模型层代码，需要对基本的数据库程序进行一定的封装和重用。以下将封装一个名为 DBConnect 的 Java 实用类，它是一个公共的类，所有需要与数据库建立连接的应用程序都只需访问并调用 DBConnect 中的有关方法即可。

8.5.1 数据库连接对象设计

DBConnect 中包含两个私有成员变量：

```
private Connection conn = null;+  
private PreparedStatement prepstmt = null;+
```

8.5.1 数据库连接对象设计

通过对以上两个私有成员变量的封装，DBConnect 可以在类的内部与数据库进行连接、执行 SQL 命令，而外部程序只需要调用 DBConnect 的相关方法即可，无需再对 Connection、PreparedStatement 对象进行具体操作。DBConnect 的主要业务方法如下：

8.5.1 数据库连接对象设计

(1) init() 方法

功能：加载连接池驱动，获得可用连接

返回类型：void

该方法的实现代码如下：

```
void init() {  
    try {  
        Class.forName("org.logicalcobwebs.proxool.ProxoolDriver");  
        conn = DriverManager.getConnection("proxool.MySQL");  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```


8.5.1 数据库连接对象设计

(2) `prepareStatement(String sql)` 方法

功能：预编译 SQL 语句

返回类型：void

该方法的实现代码如下：

```
public void prepareStatement(String sql) throws SQLException {  
    PreparedStatement pstmt = conn.prepareStatement(sql);  
}
```

8.5.1 数据库连接对象设计

(3) `prepareStatement(String sql, int resultSetType, int resultSetConcurrency)` 方法

功能：预设 SQL 语句（带 3 个参数）

返回类型：void

该方法的实现代码如下：

```
public void preparedStatement(String sql, int resultSetType,
    int resultSetConcurrency) throws SQLException {
    pstmt = conn.prepareStatement(sql, resultSetType,
        resultSetConcurrency);
}
```

8.5.1 数据库连接对象设计

(4)executeQuery() 方法

功能：执行查询 SQL 语句，返回结果集

返回类型：ResultSet

该方法的实现代码如下：

```
public ResultSet executeQuery() throws SQLException {  
    if (preparedStatement != null) {  
        return preparedStatement.executeQuery();  
    } else {  
        return null;  
    }  
}
```

8.5.1 数据库连接对象设计

(5)executeUpdate() 方法

功能：执行增加、删除和修改的 SQL 语句

返回类型：void

该方法的实现代码如下：

```
public void executeUpdate() throws SQLException {  
    if (preparedStatement != null){  
        preparedStatement.executeUpdate();  
    }  
}
```

8.5.1 数据库连接对象设计

(6)close() 方法

功能：关闭 PreparedStatement 以及 Connection 对象，释放资源

返回类型：void

该方法的实现代码如下：

```
public void close() throws SQLException {  
    if (prepstmt != null) {  
        prepstmt.close();  
        prepstmt = null;  
    }  
    if (conn != null) {  
        conn.close();  
        conn = null;  
    }  
}
```

8.5.2 DAO 设计模式

DAO (Data Access Object) 即数据访问对象，专门负责与数据库打交道，属于 MVC 中的模型层中的对象。J2EE 开发人员使用数据访问对象的设计模式，以便将低级别的数据访问逻辑与高级别的业务逻辑分离。

8.5.2 DAO 设计模式

实现 DAO 模式需要把对数据库的操作（例如增加、删除、修改等操作）全部封装在 DAO 对象里。譬如如果要插入一个新的用户记录，在 DAO 中我们只需要封装一个 `insert(User user)` 方法，具体的操作在该方法体中实现。外部程序（如 Servlet）只需要调用 DAO 中的这个 `insert(User user)` 方法即可完成整个插入新用户的操作，而无需知道插入的过程是如何具体实现的。

8.5.2 DAO 设计模式

以下将为数据库中的 Person 表构建一个对应的 DAO 类，提供对此表进行增加、删除、修改、查询等操作的实用方法。

8.5.2 DAO 设计模式

首先，在 Web Project 的 src 目录下建立名为 DAO 的子目录，在此目录下创建 DAO 接口，该接口中包含增加、删除、修改、查询等抽象方法。

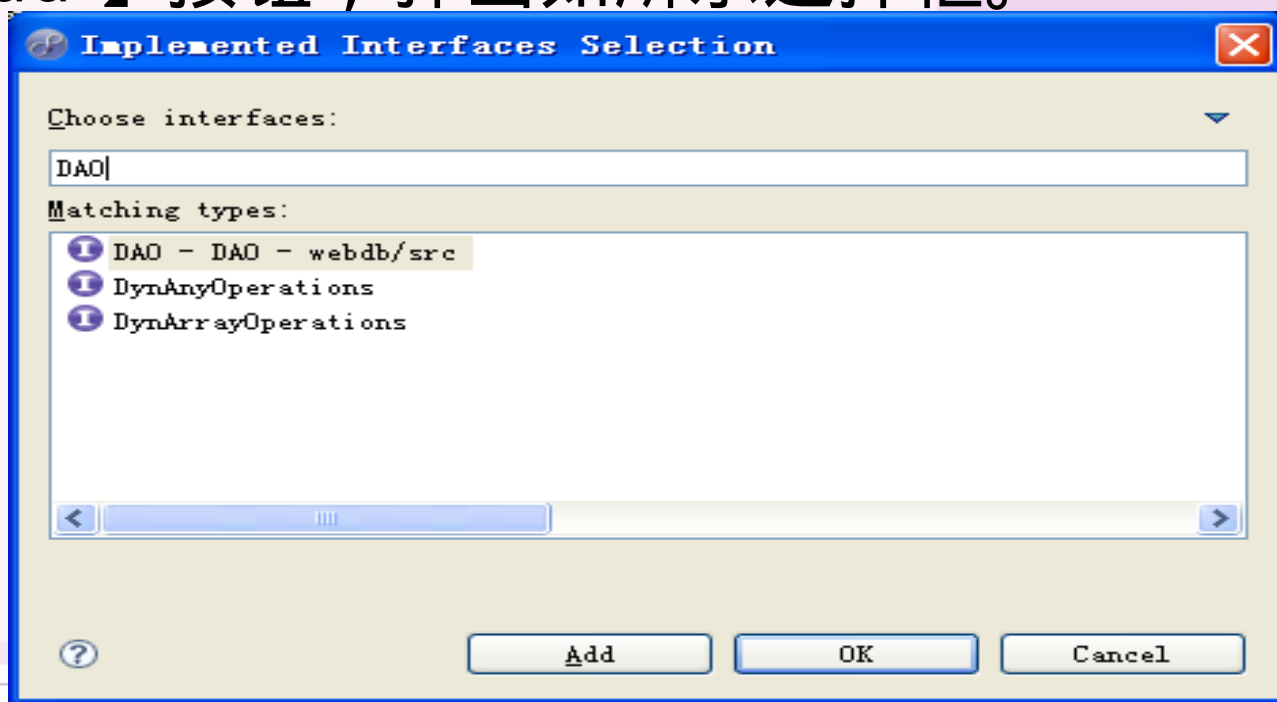
```
import java.util.List;+  
+  
public interface DAO {+  
    +  
    void insert(Object bean);+  
    void update(Object bean);+  
    void delete(Object PKey);+  
    List findAll();+  
    List findExecutingSQL(String sql, Object[] sqlParams);+  
}+
```

8.5.2 DAO 设计模式

第二步，编写 PersonDAO 类。每一个 DAO 类都实现自第一步中的 DAO 接口，按照 Java 语言的机制，这些 DAO 类中都必须实现 DAO 接口中声明的抽象方法，这样可以使得整个项目中所有的 DAO 类中增加、删除、修改、查询等方法名称都是一致的。

8.5.2 DAO 设计模式

在当前项目下创建 PersonDAO 类：在“New Java Class”向导窗体中，输入类名“PersonDAO”，单击 Interfaces 框右侧的【Add】按钮，弹出如所示选择框。



8.5.2 DAO 设计模式

在“Choose interfaces”中输入 DAO，可以看到在下方“Matching types”中列出了待选的接口，其中有我们在第二步编写好的接口 DAO，依次单击【Add】和【OK】按钮完成接口的添加，并回到类生成向导窗口。

8.5.2 DAO 设计模式

单击【 Finish 】按钮完成类的创建，看到自动生成的 PersonDAO.java 代码页

最后一步的工作是把自动生成的这些方法具体实现

8.5.2 DAO 设计模式

从以上代码我们不难看出，DAO 接口实际上提供了统一的数据操作的方法声明，所有实现该接口的操作类都要实现接口中所有的方法。所以接口是一种规范，一方面使所有的开发人员都按统一的名称、参数等进行实现和调用，另一方面使整个程序内部、程序之间的结构看起来更为清晰，有助于今后程序的重用和维护。

本章小结

本章主要介绍了 Web 开发中非常关键的数据库编程的相关内容，包括 JDBC 的基本概念和原理、JDBC 的 API 及其调用、数据库连接池以及 Web 数据库编程等技术知识。本章中的很多内容具有较强的实际操作性，例如当前热门的数据库管理系统 MySQL 的安装使用以及用 MyEclipse 开发 JDBC 程序。

8.6 本章小结

本章在最后介绍了 Web 数据库编程的主流设计模式 DAO，这也是实际 Web 项目中必须掌握的开发模式之一。在下一章的 Web 系统实例开发中将把本章的数据库编程技术运用到 MVC 模式中，作为模型层组件进行调用。

本章习题

- 1、JDBC 的驱动程序和 JDBC API 有什么联系与区别？
- 2、MySQL 的服务器和客户端分别包含哪些软件？简要描述它们各自的作用。
- 3、请在 MySQL 中建立 student 表（编号、姓名、电话、E-mail、家庭住址、邮政编码），然后编写一个 JDBC 应用程序实现对该表进行记录插入的功能。

本章习题

4. 请简述数据库连接池的基本工作原理以及 Proxool 的配置步骤。
5. 请以第 2 题中所建立的 student 表为基础，创建一个名为 student 的 JavaBean，并实现一个包含增、删、改方法的 DAO 类。